仕様記述言語

SPEC L-PERFECT

取扱説明書
「はじめにお読みください～SLP の基本的考え」編

株式会社ジェーエフピー

《はじめに》

ソフトを手にした以上は一刻も早く動かしてみたい。長い操作説明書を読むのは苦手だ。ユーザー各位の大半はそうお思いでしょうから、１ページで基本的操作ができるものをと試みました。
しかし、あえなく断念しました。このソフトは今までのものとは異なる風変わりな概念がいろいろありますので、書き始めてユーザー各位はどう操作していいか戸惑うだろうなと思ったからです。そこでこのソフトの持つ基本的な考え方を述べることにしました。途中から、ああ分かったという方は最後まで読む必要はありません。早速操作をしてみてください。画面のメニュー構成は Windows アプリケーションと同じようにしています。そして、もし分からない箇所が出てきましたら、再度この小冊子をお取りください。

目次

1. SLP の用途と特長

SLP はソフトウェアの要求仕様書を記述するためのソフトウェア・ツールです。その特徴は、簡単な文法で誰もが容易に、かつ正確に記述出来る点です。そのために自然言語を使えるようにしています。正確を期すために、記述された文書全体に渡る記述された用語のチェック、文の矛盾チェックを行なうことができます。

2. そもそも要求仕様書とは

ところで、要求仕様書はそもそもどのような文から構成されているのか。用語の説明や目次などを除けば、要求仕様書の基本は、“～せよ”という要望です。これを要求と呼ぶことにします。すると個々の要求を箇条書きにしたもの、“～すること”などの箇条書きが、要求仕様のもっともプリミティブなものといえます。さらにこれらの要求の中には、“～の条件のとき、～せよ”という条件付きのものがあります。条件がもっと複雑であったり、“せよ”という要望も複数重なったり、複雑なものもあります。

3. SLP の文法構成

SLP の文法はこのような要求仕様書の文の構造を反映したものからなっています。その文法は、考案したいくつかの文とその順序性と各文の構成要素の規則からなります。この考案した文について、その構成や他の文との関係を述べたりすることを、文の“構文”を述べているといいます。これは文の規則を語っているのですが、この規則を構文規則と呼ぶこととします。また構文規則を述べている対象の文そのものを、“構文”という場合があります。これは他の通常の文と区別するためです。

SLP で最も基本的な構文は“if-else-endif”です。この構文は、いま、ｐ、ｑ、ｒを文とすれば、この文は、“if p （ならば）、q　else（そうでないならば）、 r　endif”のような構造を表現するために使われます。文“ｐ”とは例えば、“明日は雨である”とか、“ｑ”は“(私は）読書をせよ”とか、“ｒ”は“(私は）テニスをせよ”とかを指します。

SLP ではこの表記法を

```
if p
  q
else
  r
endif
```

と、各行に文が書かれるようにしています。“endif”は構文の最後であることを示します。“else”はそうでないこと、すなわち“if”の否定を示します。よって、“else”の行には“ｐでない”ことが記述されていることになります。“ｑ”と“ｒ”は命令文です。

この“if-else-endif”構文を第１構文とすれば、“if”のないもの（従って“else”もないもの）が、単独の命令文ということになります。これを第２構文と呼ぶこととします。

すると、この第１構文と第２構文が、上に述べた要求仕様書の基本的構造と対応することに気がつきます。すなわち、要求仕様書は、その基本的な部分は、if-else-endif 構文と命令文で記述できる、といえます。命令文を“Do”構文で表現しています。

SLP の文法はこの２つの構文を基に、他の構文が考案されています。

ただし、これらの構文の記号は、ツールの環境設定で別な記号を使うことが可能になっています。ここでは、SLP が既定値として用いている表現記号を標準として各構文を説明します。

3.1　SLP の派生構文

第３、第４の構文は単位機能文、多分岐（条件）文です。それぞれ、“Fn”、“switch”で表現します。“Fn”は“Function”の略です。

単位機能文は、if-else-endif 構文や Do 構文での記述が何行にも渡った際に、記述が膨らみすぎて、if-else-endif の構造が複雑になり過ぎることを防止するための便宜的な構文です。ここで“単位”とは“ひとつのかたまり”という意味です。“最小の”単位という意味はありません。つまり、単位機能とは、記述する機能単位にまめられたものです。任意の単位機能は，さらに細かな単位機能に機能分割します。単位機能の関係は、親子関係、兄弟関係として関係づけられます。

多分岐文は、if-else-endif の派生的な文です。１つのメンバー名（後述）がいくつもの状態を持つ場合に、if-else-endif で記述するのは煩雑なので、この場合は多分岐文で表現することを可能にしています。

この多分岐文は“switch”の後ろにメンバー名を書き、メンバーの持つ様々の状態に応じて条件が分岐するように“case”の後ろに状態名を書きます。

なお“switch”構文では“else”に対応する表現を“elsw”、“endif”を“endsw”としています。

第５の構文は反復文です。“loop”で表現しています。繰り返す文の範囲を先頭の“loop”と“endlp”で包みます。“exitlp”でこの反復から抜け出ることができます。

要求仕様の箇条書きが何十行、何百行と続くものであったり、何度も変更が加えられるものであったとしたならば、箇条書きの個々の文を相互に関係づけて解析することが必要になります。SLP は if 構文の多重な表現を可能にしていますが、これらにより、複雑な要求事項を関係づけることができるようになっています。Loop 構文も複雑な表現を簡潔かつ明確にしてくれます。

第６、第７の構文は、コメント文、改行文です。前者は説明、後者は単に改行をし、文書を見やすくするためなどに用います。

3.1.1　文法の操作ガイド

SLP は文法規則を満たす必要があるため、操作ガイドに従って構文を入力していただくことになっています。Word のように自由に記述ができないので最初は戸惑うかもしれません。しかし慣れいただくと、文法に沿って記述すれば良い分むしろ楽だと感じていただけると思います。

マウスを右クリックすると、記述可能な構文、以前に記入したメンバー名、状態名、機能名が各々の入力領域で表示されます。構文は選択、項目名は選択か、自ら記述をします。皆様の自然な思考に適っていれば幸いです。

＊操作ガイド　単位機能ウインドウの Function 欄で、マウスを右クリックすると、操作ガイトとなるコンテキストメニューが表示されます。

3.2　メンバー名と状態名

ところで、if 構文などで用いられる仕様内容を記述する構文は、主語と述語に分けて記述します。これを陳述文と呼びます。命令文（Do）の場合は、目的語と具体的な命令を表す述語です。if 構文や Do 構文はこれらの用語を各々１つずつ持っておることで、陳述文、命令文を構成します。これらを合わせて単文と呼びます。

また主語と目的語の指す対象をメンバーと呼び、その名称をメンバー名と名付けます。述語（陳述述語と命令述語）の指す状態や属性を総称して状態名と呼びます。

メンバー名を“< >”でくくり、状態名を“{ }”でくくります。つまり、メンバー名、状態名はそれぞれ“< >”、“{ }”の中に記述されなければなりません。命令文は“Do”で表現しますから、例えば if-else-endif が構文として完成するのは以下のときです。

m、s はそれぞれメンバー名、状態名を指します。

```
if <m1>{s1}
  Do<m2>{s2}
else
  Do<m3>{s3}
endif
```

またここで else 行では、メンバーm1 が状態ｓ1ではない状態（s1 の否定）を指しています。なお、“でない”という否定表現を“~”で表現します。

3.2.1　時制表現

ところで、if 構文や switch 構文の状態名の後ろには、状態が「たったいま変化した」したなどのような、その時の時間を基準にした状態の変化を表現することができます。これを時制の変化と称し、その表現を助動詞で表しています。

これにより、例えば、「信号が青になった」ことと、「信号が青である」ことと、また「信号が青であった」こと、さらに「信号が青であったなら（実際は青ではなかった）」などの時制に応じた表現を区別することができます。

このような区別の必要性は、道路を渡る人（ドライバの場合でもいいのですが）がどんな状態にいるか、ということに関係して、その人の行動の具体的内容を決めるからです。

もし信号が青「になって」、そのときその人が信号を待っていたなら、歩き始めよ、ということになるでしょうし、もし信号が青「である」なら、その人は歩き「始める」までもなく、歩いてよい状態が以前から続いており、他方またその人が今たまたま信号の下で立っていたたなら歩き始めてもいいし、信号が変化する前から歩いていた人であるならば、そのままとどまらず歩き続けてよいことになります。

またこの後者の人（信号が変化する前から歩いていた人）は、ちょうどタイミングよく青「になった」ことに出くわし、止まらずに済んだなどのちょっとした状況の違いが存在します。SLP はこのようなわずかな違いも表現できるようになっています。

SLP はまた、時制を元にした状態の変化を表現し、それに応じて要求仕様を記述できるようになっています。SLP が扱う状態の変化の表現は、以下の３つです。

①　状態変化（現在の状態の変化）・・・「信号が青になった」
②　現在状態・・・・「信号が青である」
③　過去状態・・・・「信号が青であった」

上にあげた「信号が青であったなら（実際は青ではなかった）」は、SLP の範囲外とします。要求仕様の記述には、過去の出来事に反する仮定はないと考えられるからです。

また過去ではなく、現在が「信号が青であったなら（実際は現在青ではない）」も、SLP は範囲外とします。SLP では現在の状態において記述することを規則とします。この条件は、「信号は（現在）青ではない」と記述する必要があります。

とすると、過去の状態「信号が青であった」はどう解釈すべきか。これは、過去の経験を現在に持ち越した状態、といえます。信号では明確さに欠けますので、クルマの例を挙げます。「そのクルマは事故を起こした」という文は、過去のことを記述していますが、現在時制でいえば「そのクルマは事故車である」となります。このように SLP では過去の状態で表現されてとしても、現在がどうであるかと判断することになります。

また未来に関しても、①状態変化（現在の状態の変化）として解釈されます。「もし先々大地震が起きたならば」という条件は、ずっと先のあるか無いかの不確かな条件ですが、SLP では、単に「大地震が起きた」と（いつか分からぬながら）「現在の状態の変化」として扱います。要求仕様の記述がそうであるからです。

3.2.2　状態遷移

①の状態変化（現在の状態の変化）は、ある一定の並行的な状態が別な状態に変化したことを表現するものです。組込み系の世界では「状態遷移」と呼ばれるものです。状態が何重にも重なった場合には、状態の遷移を図や表などで表現することが大変つらくなる場合があります。また条件が変わる度に書き換えるのも大変です。

SLP では if-else-endif 構文で条件の重なりを表現し、特別に状態の変化するタイミングを「～となった」というような助動詞を用いることによって状態が変化する瞬間であると識別することにしています。

ソフトウェアを実装しテストを行う場合、注意しなければならない点の１つとなります。

3.2.3　状態名やメンバー名をどこまで書けばよいか

このように状態は時制と結び付いています。同じ（信号の）「青」でも、「なった」、「である」、「であった」を「青」の状態に含むと内容が異なります。

他方、状態が命令文に用いられる場合は、（信号を）青「にせよ」とか「とせよ」というように助動詞は命令形にとなっています。

SLP では記述された用語の検査を行います。記述した用語が一様に使用されているか否かを SLP 文書全般に渡りチェックします。状態名の検査においては、if 構文の状態名も Do 構文の状態名も区別なく同じようにチェックします。このとき、状態名に「青となった」と「青とせよ」が存在した場合には、SLP は異なる状態と判断します。理由は以下に述べる通りです。

用語の検査ではなく、論理矛盾のチェックの場合には、このように記述していたとしても、この違いは論理矛盾のチェックには影響を与えません。従って、この場合には if 構文と Do 構文における状態名の違いは気にすることはありません。しかし if 構文同士の場合にはきちんと統一を図って置く必要があります。SLP は現在文字列の比較でしか文字の異同を検査できませんので、語彙の異同にはご注意ください。

以上の制約を念頭に置いた上で、状態名の記述においては、「青」とか「赤」の属性や状態の記述にとどめ、if 構文の場合（や switch 構文）には助動詞を付記することをおすすめします。

なお、命令文の場合の助動詞は一様であるため、助動詞のガイドはありません。コメント文を自由に書き込むことができるようになっています。

（1）状態名やメンバー名は長く書いてかまいません

なお、SLP を実際に使用するとおわかりのように、状態名は上記のように短いものだけでははありません。またメンバー名も、単に「信号」などというケースは実際の仕様書においては稀です。実際の仕様書には、例えば「信号が青になった場合の、センサーによって検知された歩行者の数が、路上の白線より３０％以上あふれていた場合には」というように、主語（メンバー名）が、「信号が青になった場合の、センサーによって検知された歩行者の数」であり、状態名が「路上の白線より３０％以上あふれる」というように長い場合が多くあります。

ユーザー各位はこのことに、あれッと意外に思われるかもしれません。例はメンバー名の中に、重複的にコトバが幾層にも重なっています。しかし仕様書の実際はこのようなもので、このようなコトバを自然言語で記述すると、どんな事態になるか、用語の不統一となり、多様な解釈が始まり、加えてこのような重複した概念が、文字ではなく、音声で発せられた場合には、いわゆる「言った、言わない」のおぞましい風景が時に現出します。

SLP は一度記述されたメンバー名、状態名を再利用できる機能を提供しています。またコトバは一度で決まりません。コトバの背後に思考が動き、ああでもない、こうでもないと悩みます。そしてあるとき、ふと適切なコトバがひらめく場合があります。そのときは、今まで使用していた語句を SLP 文書全般に渡って一気に変更できる機能を持っています。

（2） 状態名やメンバー名を短くするために

長い用語を毎回使用するのは大変です。メンバー名や状態名を登録する機能があります（→[記述]-[メンバー名・状態名登録]。長い用語に対して、別途命名してお使いください。

（３）状態名やメンバー名の長短は？

上例の「信号が青になった場合の、センサーによって検知された歩行者の数」というメンバー名は、このメンバー名が if 構文などでより分析されるべきことを含んでいます。他方要求仕様がこのレベルの概念で提示されることも往々にしてある実態を示しています。SLP はこのような仕様の実状にあわせ、より包括的な概念で記述することも、細部にわたって分析・記述することも、いずれも可能です。

3.3　Input 欄、Output 欄

if 構文や switch 構文に記述されるメンバー名と状態名は、単位機能ウインドウの Function (記述)欄においては、左の Input 欄の同じ行に表示されます。単位機能は通常複数行からなりますから、その一覧はどんな Input 情報が単位機能の要件になっているのかを明示します。用語の関係をリアルタイムに修正することができるため、このようなことが可能なのです。

右の Output 欄は、命令文（Do 構文）によって、メンバー名の状態が変更されたことを示します。これも Inputu 欄と同じく、メンバー名、状態名の全文書に渡る関係をリアルタイムに把握し、修正するためです。

Input 欄と Output 欄をまとめて、単位機能の IO 部と呼びます。

なおこの欄は[ツール]-[環境設定]メニューで表示、非表示の設定ができます。

4.　目次

4.1　階層順目次

階層順目次とは、目次を階層的に表現したものを指します。階層的に表現された目次の１つは、既に述べた単位機能に相当します。単位機能は単位機能を記述する画面ばかりでではなく、この階層目次でも記述することができます。ただし、機能のボックスと機能名だけです。単位機能の詳細な記述は、単位機能を記述する画面で行ないます。

階層順目次では子や兄弟の単位機能を追加したり削除したりすることができます。ただし、ここで（階層順目次ウインドウ）で作成したこれらの単位機能ボックスは、単位機能を記述する画面（単位機能ウインドウ）で使用されない限り、組込まれていない単位機能だとみなされます。これを「未組込み」単位機能と呼びます。

「使用される」とは、単位機能を記述する画面で、“Fn”として、単位機能ボックスと同じ機能の名前が記述されたとき、その単位機能ボックスは正式の Fn となったことを指します。すなわち、両者は同じものとしてコンピュータは判定し、単位機能ボックスを Fn として「組込んだ」とします。

なお、ID 番号は、起点単位機能（最上位）の ID の番号を“0”としています。第２の階層では、ID 番号を 1、2、3、…n、と自動採番します。以下は間に“.”を付加して表現します。

階層目次における単位機能ボックスの作成や削除の実際は、操作説明書を参照してください。

4.2　ID 順目次

階層順目次に対して、縦に ID 番号順に並べたものを ID 順と呼びます。操作は階層順目次と同じです。

4.3　単位機能の既記述と未記述

単位機能には、組込み、未組込みの他に、既記述、未記述という概念があります。機能名を記述した場合、単位機能は既記述とします。機能名を記述していない場合は、単位機能は未記述とします。これらを単位機能の発生（作成）に応じて整理します。なお単位機能、Fn、ID はここでは同義とします。“ID”という呼称は目次（階層順、辞書順）を作成する際の慣用から来ています。

（１）　**単位機能作成時**

単位機能作成画面（Function 欄）で Fn を記述し、かつ Fn の中に機能名を記述したとき、その Fn は既組込み、かつ既記述となります。

（２）　**階層順（辞書順）目次作成時**

階層順（辞書順）目次作成画面にて、単位機能ボックスを発生（作成）させ、かつ機能名を記述するとき、機能名を書いているので、その単位機能ボックスは未組込み、既記述となります。

以上を表にまとめます。

単位機能を Function 欄で記述する			単位機能を目次欄で作成する		
Fn 記述	機能名記述	単位機能の状態	ﾎﾞｯｸｽ発生	機能名記述	単位機能の状態
○	○	既組込＆既記述	○	○	未組込＆既記述
○	×	既組込＆未記述	○	×	未組込＆既記述
×	○	発生しない	×	○	発生しない
×	×	発生しない	×	×	発生しない

（３）　**未組込み単位機能の組込み化**

目次欄で作成した単位機能は、単位機能作成画面（Function 欄）で同じ機能名を持ったとき、即ち同じ機能名が記述されたとき既組込み化されます。目次欄の単位機能が機能名を持っていないときは、組込まれることはありません。

5.　単位機能のヘッダー部の項目

単位機能は、Functuoin（記述）欄で、機能を SLP 構文で記述しますが、付加する情報として以下の項目があり記述できるようになっています。まとめて、単位機能のヘッダー部と呼びます。

（１）　**機能名**

当該の単位機能に命名するためのものです。ユニークでなければなりません。

（２）　**機能内容**

単位機能の機能内容を記します。オブジェクトのリンクと埋め込み（OLE）に対応していますので、図や表等で補足できます。

（３）　**ＩＤ**

当該単位機能をアイデンティファイするものです。自動採番されます。

（４）　**機能属性**

当該単位機能に任意の属性を記すことができます。非機能要件とか、機能安全に関するものとか、情報セキュリティに関するものとか、あるいはプロジェクト固有の注意事すべき事項を盛り込むことができます。未記入可です。

（５）　**原要求項目**

当該単位機能に対応する要求仕様項目です。要求仕様項目を識別するアルファベットなどの記号等で使用します。未記入可です。

（６）　**当工数**

当該工数に関する工数を記入できます。上位の単位機能に集計されます。未記入可です。

6.　環境設定項目

文字の装飾、構文記号、ＩＤの記号などを設定できます。[ツール]-[環境設定] メニューにあります。

《おわりに》

以上基本的な事柄を述べました。早速 SLP をお試しください。
なお、不明点については下記宛にメールでお問い合わせください。
E-mail: slp-support@jfp.co.jp